

Implementation of 32-Bit RISC Processor using Reversible Gates

Jyoti Choudhary¹, Mahesh Kumar Sharma²

¹PG Scholar, ²Assistant Professor

^{1,2}Rajasthan College of Engineering for Women, Jaipur, Rajasthan, India

How to cite this paper: Jyoti Choudhary | Mahesh Kumar Sharma "Implementation of 32-Bit RISC Processor using Reversible Gates" Published in

International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-3 | Issue-5, August 2019,



IJTSRD26709

pp.1789-1793,

<https://doi.org/10.31142/ijtsrd26709>

Copyright © 2019 by author(s) and International Journal of Trend in Scientific Research and Development Journal. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (CC BY 4.0) (<http://creativecommons.org/licenses/by/4.0>)



In those technologies the CMOS gates are better than the bipolar gates. But the CMOS gates are having its own limitations for that reason we are moving to reversible gates. Reversible logic has received great attention in the recent years due to their ability to reduce the power Dissipation which is the main requirement in low power VLSI design. It has wide applications in low Power CMOS and Optical information processing, DNA computing, Quantum computation and nanotechnology. High-performance chips releasing large amounts of heat impose practical limitation on how far can we improve the performance of the system. Reversible circuits that conserve information, by uncomputing bits instead of throwing them away, will soon offer the only physically possible way to keep improving performance. Since the development of the stored-program computer, there has been remarkably few true innovations in the areas of computer organization and architecture. In this list, one of the most interesting and potentially one of the most important innovations is Reduced Instruction Set Computer (RISC) architecture. The RISC architecture is a dramatic departure from the historical trend in CPU architecture and challenges the conventional wisdom expressed in words and deeds by most computer architects. The aim of the project is to:

- Design the architecture of simple 32- Bit RISC processor.
- Implementation of individual modules of the RISC processor using VHDL.
- Integration of individual modules.
- Their Simulation and Synthesis.

ABSTRACT

Reversible logic is one of the emerging technologies having promising applications in quantum computing. The aim of this project is to design the schematic and simulation for a 32-bit RISC processor using reversible logic peres gate. Beside the functional development, by optimizing the speed of our processor in every block which is inside that, and to minimize the overall delay conventional gates are replaced with reversible gates. This reversible gates which are applicable in Nano technology, Quantum computing, Low power CMOS, Optical computing. This RISC embodies 15 basic instructions involving Arithmetic, Logical, Data Transfer and control instructions. To implement these instructions the design incorporates various design blocks like Control Unit (CU), Arithmetic and Logic Unit (ALU), Accumulator, Program Counter (PC), Instruction Register (IR), Memory and additional logic. Design is implemented and verified in VHDL in Xilinx 14.3.

KEYWORDS: RISC, Arithmetic Logic Unit, CISC, Reversible Gates

I. INTRODUCTION

In VLSI which is the growing technology in our day to day life. Everyday new emerging technologies are introducing in market. So the technology development is the most important in our current life. In past the computer occupied the entire room. Gradually the size has reduced and today come to the level of hand based equipment's. At first in any device is made by the IC which is based on the conventional gates. In VLSI it having the ultimate goal is less area, high speed, and low power.

A CISC system offers a large menu of features which implies a larger and more complicated decoding subsystem and complex logic which is not in the case of RISC Processors. One of the ways to increase the speed of execution on any computer is to implement Pipelining which is difficult to implement in CISC systems. But in RISC processors Pipelining can be efficiently implemented. Due to this opposition to the traditional CISC design, in early eighties there emerged a new trend of computer design, i.e. RISC, which became popular thereafter. Several advantages of RISC processors are:

- VLSI realization
- Increase in computing speed
- Less time to complete the design
- Reduction in overall design cost
- High Level Language (HLL) support

RISC processors are primarily used in work stations and personal computers. RISC processors with integrated graphics and display systems can be used in car navigation systems, driver information systems etc. It is also applicable in high speed data transmission and real time controlling applications.

II. REVERSIBLE GATES

A function is reversible if each input vector produces a unique output vector. Reversible logic is of growing importance to many future computer technologies. It is not possible to realize quantum computing without implementation of reversible logic. The main purposes of

designing reversible logic are to decrease quantum cost, depth of the circuits and the number of garbage outputs. The reversible circuits form the basic building block of quantum computers as all quantum operations are reversible. The two main constraints of reversible logic circuits are:-

- Fan-out not possible.
- Feedback or loops not allowed.

A reversible logic circuit should have the following salient features:

- Use minimum number of reversible gates.
- Use minimum number of garbage outputs.
- Use minimum number constant inputs.

A. NOT Gate

The simplest Reversible gate is NOT gate and is a 1*1 gate. The Reversible 1*1 gate is NOT Gate with zero Quantum Cost is as shown in the Figure 1.1. If input is A then output is $P=A$



Fig.1. NOT Gate

B. Feynman Gate (FG)

Feynman gate is a 2*2 one through reversible gate as shown in figure 2. The input vector is $I(A, B)$ and the output vector is $O(P, Q)$. The outputs are defined by $P=A$, $Q=A \oplus B$. It is the universal reversible gate.



Fig.2. Feynman Gate

C. Peres Gate (PG)

It is a basic reversible gate which has 3-inputs and 3- outputs having inputs (A, B, C) and the mapped outputs $(P=A, Q=A \oplus B, R=(A.B)^C)$. This PG is depicted in Figure.3

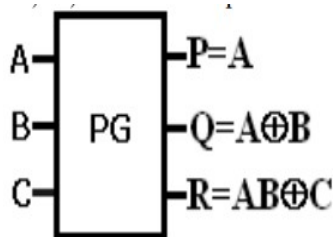


Fig.3. Peres Gate

III. PROPOSED RISC PROCESSOR

The design of 32-bit RISC processor incorporates various design blocks like Arithmetic Logic Unit (ALU), Accumulator, Program Counter (PC), Instruction Register (IR), Memory, Control Unit (CU), and additional logic.

The design incorporates some the following issues.

- It handles 32 bit data ,28 bit address
- Uses fixed instruction format of length 32 bit.
- Size of opcode is of 4 bit, handling 15 instructions.
- Has 256 memory locations
- 32-bit registers (IR,ACC)
- Implements 2-staged pipelining i.e overlaps of fetch and execute cycles.

- Has two addressing modes ,Register addressing and memory addressing modes
- No interrupts and No conditional branches
- Data that it handles is unsigned integer type.

The next sections of this chapter presents the schematic diagrams of individual modules, explanation of their function, flow chart to implement the function, VHDL description of the individual modules. In the last section the top order module Block diagram, functioning, VHDL description of the top module are given.

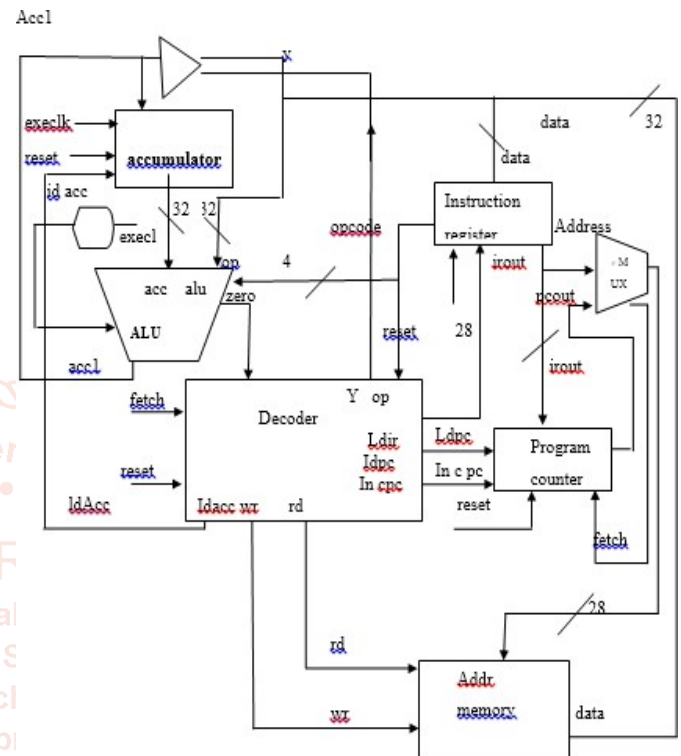


Fig.4. Proposed RISC Processor

A. ALU

The ALU performs both arithmetic and logical operations and as well as control of transfer instructions. It takes data and acc as inputs to generate output according to the opcode. An execlk is given as input for synchronization and the output is available at positive edge of the execlk. It performs arithmetic and logic instructions directly and control of transfer instructions are performed with the help of control and logic decoder. The arithmetic-logic unit (ALU) performs all arithmetic operations (addition, Subtraction, multiplication, and division) and logic operations. Logic operations test various conditions encountered during processing and allow for different actions to be taken based on the results.

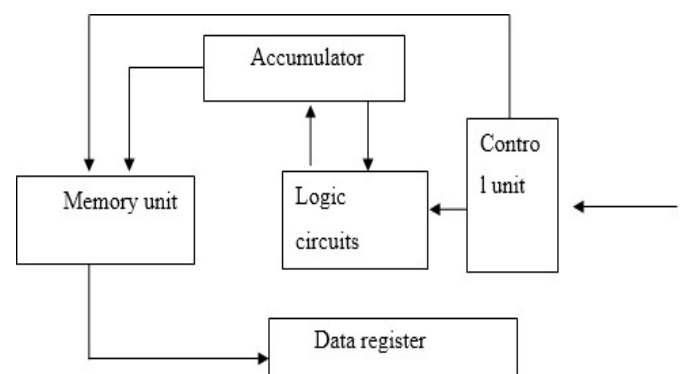


Fig.5. Block Diagram of ALU

The data required to perform the arithmetic and logical functions are inputs from the designated CPU registers and operands. The ALU relies on basic items to perform its operations. These include number systems, data routing circuits (adders/subtractors), timing, instructions, operands, and registers. Figure shows are presentative block diagram of an ALU of a microcomputer.

B. ACCUMULATOR

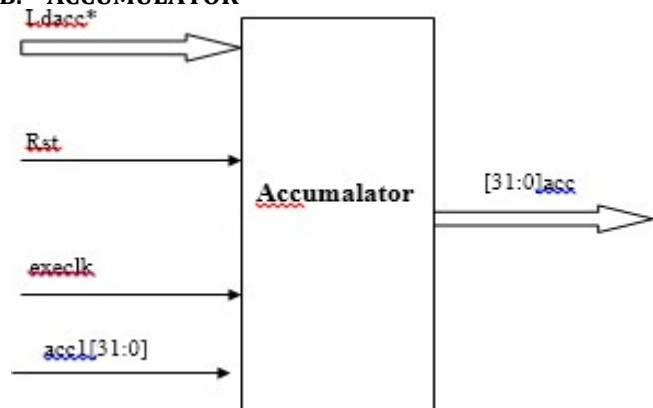


Fig.6. Block Diagram of an Accumulator

The result of an Alu operation is always stored in accumulator at some specified time based on the control logic instruction and also the execclk. This output is again fed to Alu as input. If Reset =0, the output of accumulator is cleared to zero. When reset is high and load accumulator signal is set high, the output of the ALU is loaded in to the accumulator at the neg.edge of the execlock.

C. Buffer

Used for writing data in to memory. When it is required to write data in to the memory, then necessary control signals are generated at the buffer. Buffer is used for achieving bi-directional operation of the data bus.

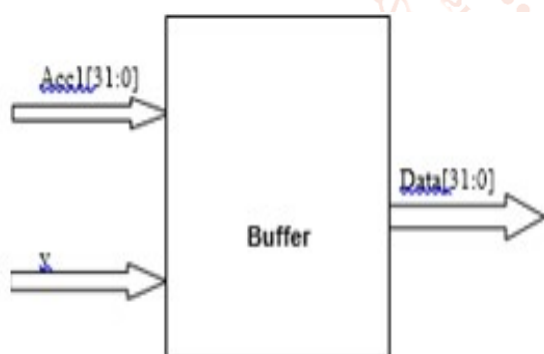


Fig.7. Block Diagram of Buffer

D. Memory

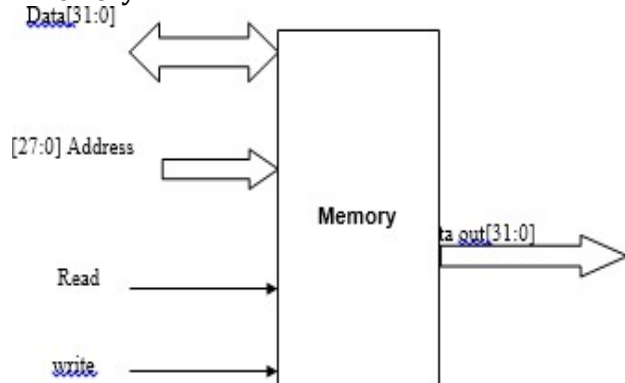


Fig.8. Memory Unit

TABLE I Operation of Memory Unit

read	Write	operation
0	0	No operation
0	1	Write operation
1	0	Read operation

TABLE II Functioning of Memory

Inputs	Size(in Bits)	Description
Rd	1	Enables the output buffer of the Memory registers and makes the data available on the Data line when Rd=1. Remains in high impedance state when Rd=0.
Wr	1	Enables the input buffer of the Memory registers and makes the data available on the Data line when Wr=1. Remains in high impedance state when Wr=0.
Addr	28	Carries the address of the Memory location to be Read or Written upon.
InOut	Size(in Bits)	Description
Data	32	Bi-directional bus that carries all the data from & to the Memory or Instruction register or Accumulator.

E. Multiplexer

Multiplexer provides memory access to either IR (instruction register) or PC (program counter) based on Fetch signal. Memory has both data and instructions. In order to access both data and instructions through a single address port a multiplexer is needed. It selects address according to the fetch signal. If the fetch signal is high pcout is selected and irout is selected when fetch signal is low.

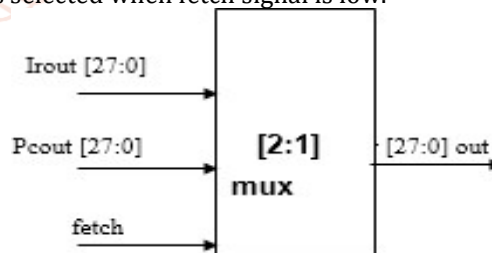


Fig.9. Multiplexer

F. Program Counter

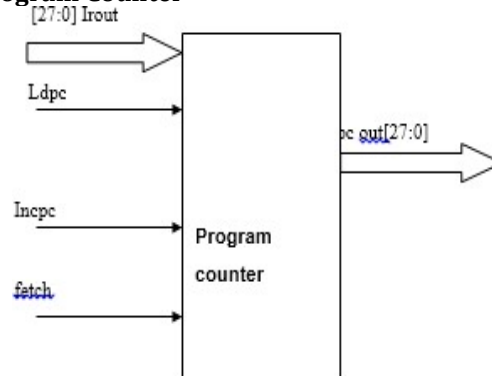


Fig.10. Block Diagram Counter of Program Counter

The program counter (PC) contains the address of the next instruction. The CPU fetches an instruction from memory, executes it and increments the content of PC. Thus in the next instruction cycle, it will fetch the next instruction of the program pointed out by the program counter. The instructions are executed sequentially unless an instruction changes the content of program counter.

G. Instruction Register

Instruction register is a 32-bit register which gets loaded with data from the memory. When LdIr signal is high, the data bus contents get loaded into the Instruction register. The 4-MSB's of this loaded data [31:28] are the OpCode of the Instruction and the remaining bits [27:0] are the address of a memory location to fetch the subsequent data. On the falling edge of the asynchronous signal Rst the IR gets cleared irrespective of any condition. Outputs of IR module are IrOut (lower 28-bits) and OpCode (upper 4-bits) of the data fetched from the memory.

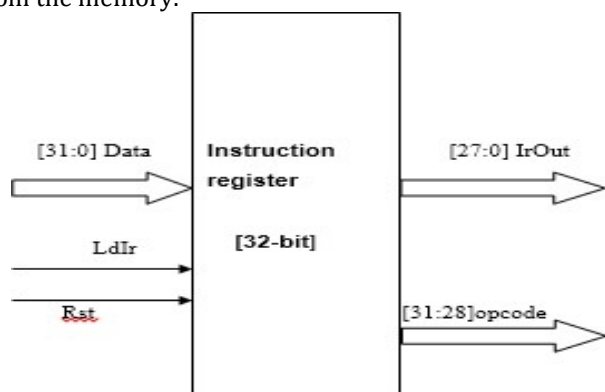


Fig.11. Block Diagram of Instruction Register

H. Control Unit

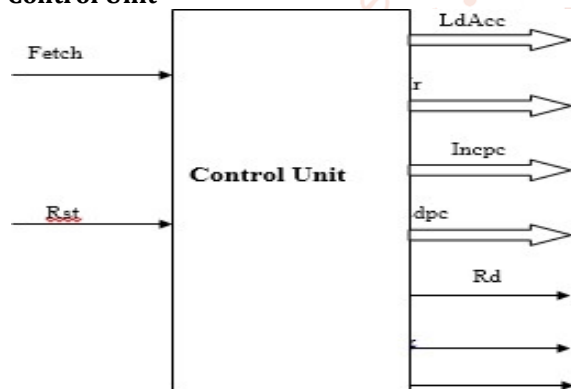


Fig.12. Block Diagram of an Control Unit

The control logic generates all the necessary control signals required for satisfactory operation of the CPU. The control signals are load accumulator, load instruction register, increment program counter, load program counter, memory read, memory write. When LdAcc is set high, the output of the ALU is loaded into the accumulator. When Ldpc is set high program counter is loaded with the address from where the next instruction is to be fetched. When Ldir is set high the instruction register is loaded with the instruction that is to be executed. If increment program counter signal is set high, program counter is incremented by one to the previous value. If write signal is set high and read low then the data bus is being written on the address of the memory location indicated by the program counter. If read signal is high and write low this is memory read operation and the data at the memory address indicated is placed on the data bus.

IV. RESULTS

The simulation and synthesis result for code for 32-bit- RISC design written in VHDL and verified in Xilinx is shown in below figures

Synthesis Results

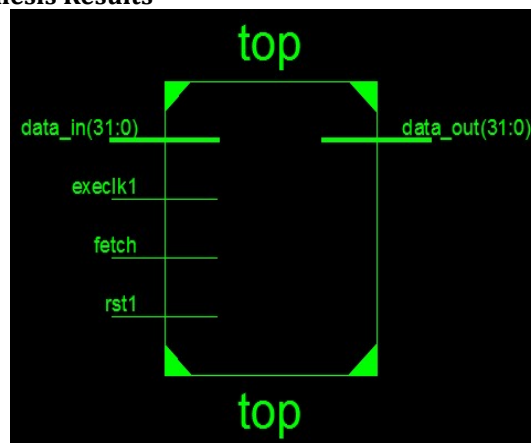


Fig.13. Block Diagram

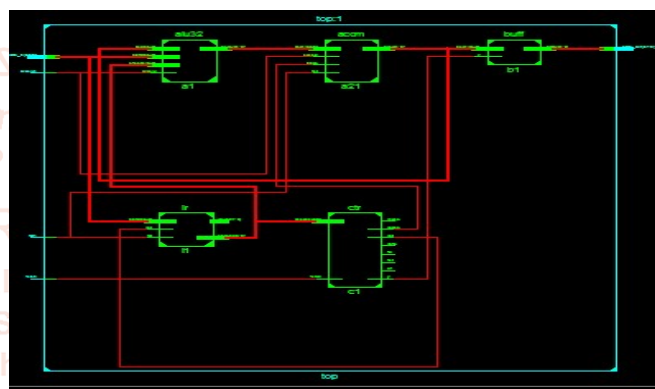


Fig.14. RTL Schematic Diagram

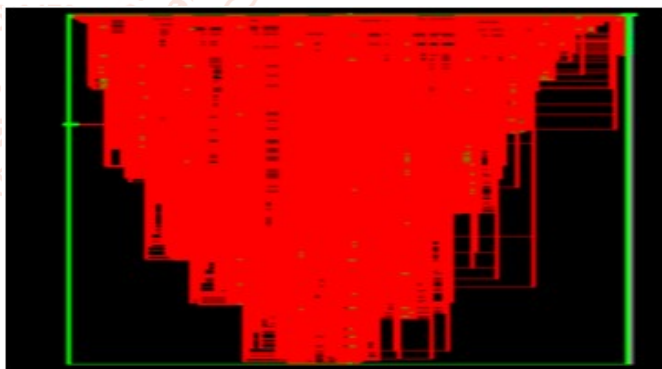


Fig.15. Technology Schematic

V. SIMULATION WAVEFORMS

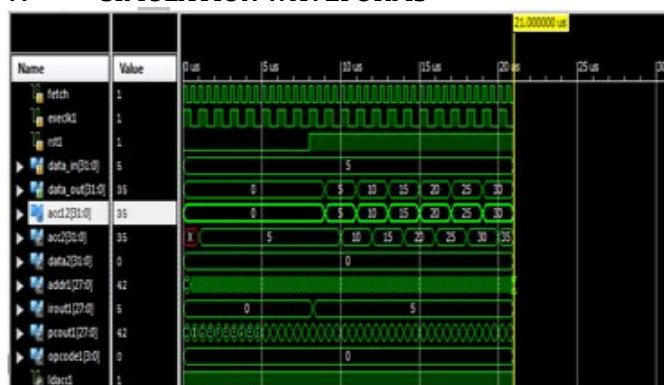


Fig.16. Addition

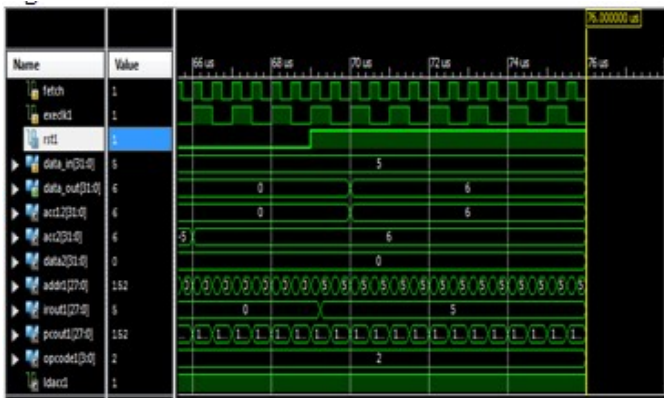


Fig.17. Increment

VI. CONCLUSION

VII. REFERENCES

- [4] Y. symala & A. V. N Tilak , "Reversible Arithmetic logic unit", IEEE conference, Page no: 207-211, 2009
- [5] D. Mandalidis, P. Kenterlis, J. Ellinas, "A computer architecture educational system based on a 32-bit RISC processor," International Review on computers and Software, pp. 114-119, 2008.
- [6] Antonio H. Zavala, Jorge Avante R., "RISC- Based Architecture for computer Hardware Introduction"
- [7] M. Jaumain, M. Osee, A. Richard, A. Vander Biest, P. Mathys, "Educational simulation of the RiSC processor," ICEE International Conference on Engineering Education, 2007.
- [8] David A. Patterson, John L. Hennessy, "Computer Organization and Design - The Hardware/Software Interface" Second Edition (1998) Morgan Kaufmann Publisher, Inc.
- [9] Xiao Li, Longwei Ji, Bo Shen, Wenhong Li, Qianling Zhang, "VLSI implementation of a High-performance 32-bit RISC Microprocessor", Communications, Circuits and Systems and West Sino Expositions, IEEE 2002 International Conference on, Volume 2, 2002 ,pp. 1458-1461.
- [10] Kusumlata Pisda, Deependra Pandey, "Realization & Study of High Performance MIPS RISC Processor Design Using VHDL", International Journal of Emerging trends in Engineering and Development, Volume 7, Issue 2, November 2012, pp. 134 – 139, ISSN: 2249 – 6149.
- [11] Kirat Pal Singh, Shivani Parmar, "VHDL Implementation of a MIPS – 32 bit Pipeline Processor", International Journal of Applied Engineering Research, Volume 7, Issue 11, ISSN: 0973 – 4562.
- [12] Samiappa Sakthikumaran, S. Salivahanan and V. S. Kaanchana Bhaaskaran, "16-Bit RISC Processor Design For Convolution Application", IEEE International Conference on Recent Trends In Information Technology, June 2011, pp.394-397.
- [13] Rupali S. Balpande and Rashmi S. Keote, "Design of FPGA based Instruction Fetch & Decode Module of 32-bit RISC (MIPS) Processor, International Conference on Communication Systems and Network Technologies pp. 409 –413.
- [14] R. Uma, "Design and Performance Analysis of 8 – bit RISC Processor using Xilinx Tool", International Journal of Engineering Research and Applications, Volume 2, Issue 2, March – April 2012, pp. 053 – 058, ISSN: 2248 – 9622.
- [15] V. N. Sireesha and D. Hari Hara Santosh, "FPGA Implementation of a MIPS RISC Processor", International Journal of Computer Technology and Applications, Volume 3, Issue 3, pp. 1251 – 1253, ISSN: 2229 – 6093.
- [16] M. Yugandhar and N. Suresh babu, "VLSI Design of Reduced Instruction set Computer Processor Core Using VHDL", International Journal of Electronics, Communication & Instrumentation Engineering Research and Development (IJEIERD), Volume 2, Issue 3, pp. 42 – 47, ISSN: 2249 – 684X.